

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ - ПРОЦЕССОВ
УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ И
МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Матвеев Владимир Валерьевич

Выпускная квалификационная работа бакалавра

Контроль сохранности лесоматериалов при
транспортировке с использованием
компьютерного зрения

Направление 010400

Прикладная математика и информатика

Заведующий кафедрой,
доктор физ.-мат. наук,
профессор Андрианов С. Н.

Научный руководитель,
доц. Гришкин В. М.

Рецензент:
проф. Матросов А. В.

Санкт-Петербург
2016

Содержание

Введение	3
Постановка задачи	5
Обзор литературы	6
Глава 1. Структура алгоритма распознавания.	7
1.1. Предобработка	7
1.2. Поиск окружностей на изображении	8
1.3. Постобработка. Шаг 1	10
1.4. Постобработка. Шаг 2	13
1.5. Постобработка. Шаг 3	14
Глава 2. Используемые алгоритмы	17
2.1. Приведение цветного изображение к полутоновому	17
2.2. Медианный фильтр	18
2.3. Свертка изображения с ядром	19
2.4. Фильтр Гаусса	19
2.5. Метод гамма-коррекции	21
2.6. Детектор границ Канни	23
2.7. Алгоритм Хафа для поиска окружностей	24
2.8. Цветовое пространство LAB	26
2.9. Сегментационный фрактальный анализ текстур	28
2.10. Метод опорных векторов	30
2.11. Кластеризация методом k-средних	31
2.12. Алгоритм DBSCAN	32
Глава 3. Программная реализация и результаты тестирования	35
Выводы	37
Заключение	38
Список литературы	39
Приложение	41

Введение

В ходе работы лесозаготовливающего предприятия возникает необходимость вести контроль качества перевозки древесины от мест вырубki до мест переработки. Это подразумевает под собой выявление фактов пропажи и подмены бревен в момент транспортировки. Данный процесс предлагается организовать следующим образом: делаются фотографии лесовоза сразу после загрузки и перед разгрузкой, эти фотографии сравниваются и по ним определяется наличие факта подмены или пропажи лесоматериалов в пути. Сравнение происходит за счет выявления и сопоставления особенностей торцов бревен. Программное обеспечение решающее эту задачу должно первым шагом находить торцы бревен на фотографиях, а затем сравнивать их.

В данной работе предложен алгоритм для решения первой части этой задачи, то есть для нахождения торцов бревен на изображении. Следует учитывать, что по одной фотографии анализируются только одна пачка бревен. Если на изображении видны бревна, не относящиеся к анализируемой пачке, то их не следует находить. Кроме того, алгоритмы анализа изображений не могут работать с абсолютной точностью, поэтому предполагается, что работать данная программа будет под контролем оператора, который будет иметь возможность исправлять ошибки. При этом работа по устранению этих ошибок будет занимать значительно меньшее время, чем сравнение фотографий в ручную. Таким образом, производительность человеческого труда существенно возрастет. Пример входного изображения приведен на рисунке 1.



Рис. 1: пример входного изображения

Постановка задачи

На вход алгоритму подается цветное растровое изображение \mathbf{I} в цветовом пространстве RGB. Предполагается, что на нем изображена задняя часть загруженного лесовоза таким образом, что видны торцы лежащих в кузове бревен. Для нахождения торцов бревен будем приближать их окружностями. Таким образом, на выходе из программы будем иметь массив \mathbf{A} размерности $3 \times n$, где n – количество обнаруженных торцов. Например, $\mathbf{A}[1, i]$ – радиус окружности с номером i , $\mathbf{A}[2, i]$, $\mathbf{A}[3, i]$ – абсцисса и ордината центра окружности с номером i . Эти окружности и будут задавать положение искомых торцов бревен на изображении.

Обзор литературы

При написании данной работы было использовано множество алгоритмов и методик, описанных в различной литературе. Особенно стоит отметить книгу [6]. Это одна из наиболее популярных книг по методам машинного обучения. В ней подробно описаны многие методы, нашедшие применение в данной работе. В этой работе рассматриваются такие методы, как алгоритм k-средних, SVM и кросс-валидация. Материал этой книги был чрезвычайно полезен при написании данной работы.

Также необходимо упомянуть статью [5], в которой описан алгоритм SFTA, примененный при извлечении текстурных признаков из изображений. Данный алгоритм, разработанный и описанный авторами статьи, позволил существенно улучшить качество решения задачи классификации изображений. В работе приведена схема устройства алгоритма, описаны его преимущества и области применения. Также в работе описан многоуровневый алгоритм Отцу, применяемый в SFTA для сегментации изображений.

В статье [1] изложена модификация алгоритма Хафа, предназначенная для поиска на изображении окружностей неизвестного радиуса. Использование данной модификации алгоритма существенно улучшает качество распознавания искомых объектов. Эта статья несёт в себе подробную информацию по настройке и использованию алгоритма Хафа. Она стала одной из ключевых при написании данной работы.

Глава 1. Структура алгоритма распознавания

1.1. Предобработка

Под предобработкой понимается улучшение качества изображения в соответствии с целями, обусловленными дальнейшим алгоритмом обработки изображения. В данной работе на этапе предобработки изображение переводится в полутоновое и к нему применяются различные методы фильтрации для удаления шумовых помех. Будем обозначать полутоновое изображение I_g . Это изображение будем использовать при поиске окружностей, при этом оригинальное изображение I еще понадобится на этапе постобработки.

Алгоритм предобработки выглядит следующим образом. Изображение прошедшее предобработку приведено на рисунке 2.

- Построение полутонового изображения I_g .
- Применения медианного фильтра [9] для удаления помех типа "соль перец".
- Применение фильтра Гаусса [4] для удаления остаточного шума. Размерность ядра фильтра взята 5×5 , $\sigma = 10$
- Контрастирование изображения I_g методом гамма-коррекции [7].

Пример изображения прошедшего предобработку приведен на рисунке 2.



Рис. 2: изображение прошедшее предобработку

1.2. Поиск окружностей на изображении

Поиск окружностей осуществляется с помощью алгоритма, базирующегося на преобразовании Хафа [2] [1]. Это численный метод позволяющий находить на изображении графические примитивы, принадлежащие определённому классу фигур. В том числе это могут быть окружности. На вход алгоритму подаются следующие параметры.

- Изображение I_g .
- Значения минимального и максимального радиуса искомых окружностей.
- Порог чувствительности алгоритма Хафа.
- Порог бинаризации детектора Канни, который используется для нахождения контурного изображения перед применением преобразования Хафа.

Параметры алгоритма подбирались в ручную на основании свойств имеющихся изображений. Для имеющихся данных это сделано следующим образом:

- Значения минимального и максимального радиуса равны 20 и 80 пикселей соответственно.
- Порог чувствительности алгоритма Хафа равен 0.965.
- Порог бинаризации детектора Канни равен 0.1.

Алгоритм возвращает массивы в которых записана информация о радиусах и координатах центров окружностей, содержащихся на изображении.

Далее по полученным данным и изображению \mathbf{I} строится массив \mathbf{C} размерности $1 \times m$, где m – количество найденных окружностей. Элемент массива представляет собой структуру со следующими полями.

- Радиус окружности r .
- Абсцисса центра окружности x .
- Ордината центра окружности y .
- Фрагмент \mathbf{I}_f исходного изображения I , содержащий данную окружность.

Ниже представлен код MATLAB для получения такого фрагмента изображения.

```
If= I(floor(max(y - r, 1)):floor(min(y + r, h)), ...  
      floor(max(x - r, 1)):floor(min(x + r, w)), :);
```

В данном фрагменте кода h , w - высота и ширина изображения соответственно. Изображение с найденными на нем окружностями приведено на рисунке 3.

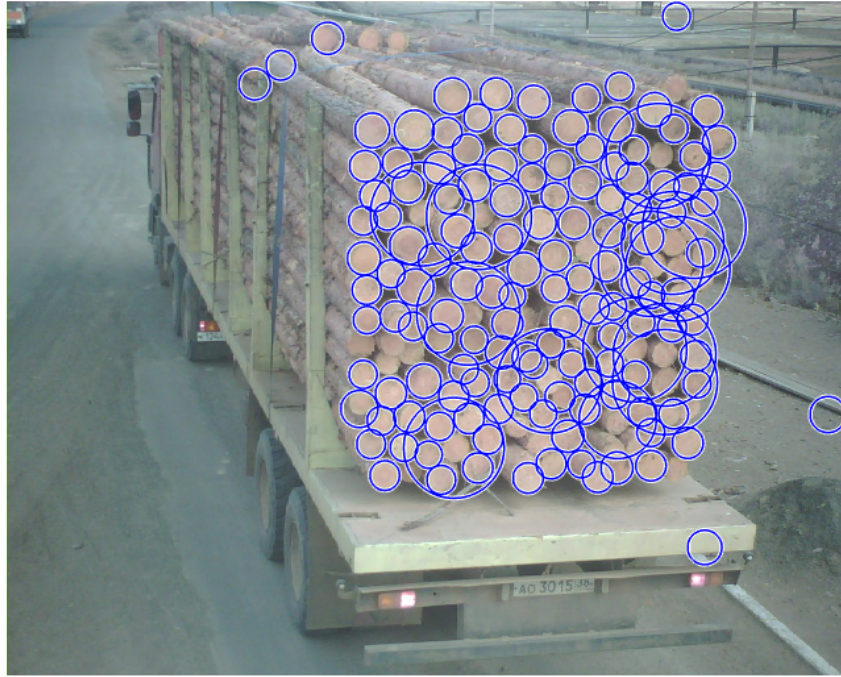


Рис. 3: исходное изображение с найденными на нем окружностями

1.3. Постобработка. Шаг 1

В ходе постобработки осуществляется проверка, какие из найденных на предыдущем шаге окружностей содержат торцы бревен, а какие нет. Эта задача решается с помощью методов машинного обучения на основании цветовых и текстурных признаков.

На входе имеем массив \mathbf{C} из предыдущего параграфа. Исходя из постановки задачи можно предполагать, что большинство из его элементов соответствуют искомым объектам - торцам бревен. Однако, необходимо исключить окружности, которые являются элементами фона и их надо удалить из массива \mathbf{C} . Для преодоления этой проблемы решим задачу машинного обучения с учителем [6] на объектах \mathbf{I}_f . С помощью метода опорных векторов с линейным ядром [6] построим алгоритм бинарной классификации, где метка 1 означает, что \mathbf{I}_f содержит торец бревна, а 0 - не содержит. Размер обучающей выборки составляет 2000 объектов. Получим массив \mathbf{C}_1 , из которого удалены окружности, не классифицированные как торцы бревен. Результат данного шага по-

стообработки можно увидеть на рисунке 4.

Далее рассмотрим признаковое представление изображений \mathbf{I}_f . Будем выделять цветовые и текстурные признаки этих изображений. Представим \mathbf{I}_f в цветовом пространстве LAB и построим гистограммы H_a , H_b длиной 64 элемента для измерений A и B. Уменьшение длины векторов гистограмм (с 256 до 64) сделано с целью понижения общей длины вектора признаков и, тем самым, снижения вычислительных затрат при обучении классификатора. Переход от цветового пространства RGB к LAB обусловлен необходимостью обеспечить инвариантность алгоритма относительно яркости изображения. В цветовом пространстве RGB при изменении яркости пикселя изменяется каждая из координат, описывающих его. Этого не происходит в пространстве LAB, где координата L отвечает за яркость, а A и B от яркости не зависят.

С помощью алгоритма сегментационного фрактального анализа SFTA [5] получим вектор D , описывающий текстуру изображения \mathbf{I}_f . Параметр алгоритма n_t , отвечающий за количество пороговых значений рекурсивного алгоритма Отцу [5], который используется при применении данного метода, выбран равным 4. На выходе получим вектор размерности 32×1 , характеризующий текстуру входного изображения. Конкатенация векторов H_a , H_b , D будет являться вектором признаков изображения I_f , размерность такого вектора равна 160×1 .



Рис. 4: исходное изображение с найденными на нем окружностями, выполнен шаг постобработки 1

Покажем состоятельность построенных признаков. С помощью метода t-SNE (t-distributed stochastic neighbor embedding) [12] понизим размерность признакового пространства для обучающей выборки до 2. Этот алгоритм применен в соответствии с рекомендациями использовать его вместе с методом главных компонент [6]. То есть если размерность признакового пространства больше 30, то сначала применяется метод главных компонент, чтобы понизить размерность до 30. Затем к полученным данным применяется метод t-SNE. Теперь, отобразив обучающую выборку на плоскости, можно заключить, что классы хорошо разделимы. Визуализация данных представлена на рисунке 5. Действительно, оценка ошибки построенного классификатора, рассчитанная с помощью метода скользящего контроля [6], равна 0.04.

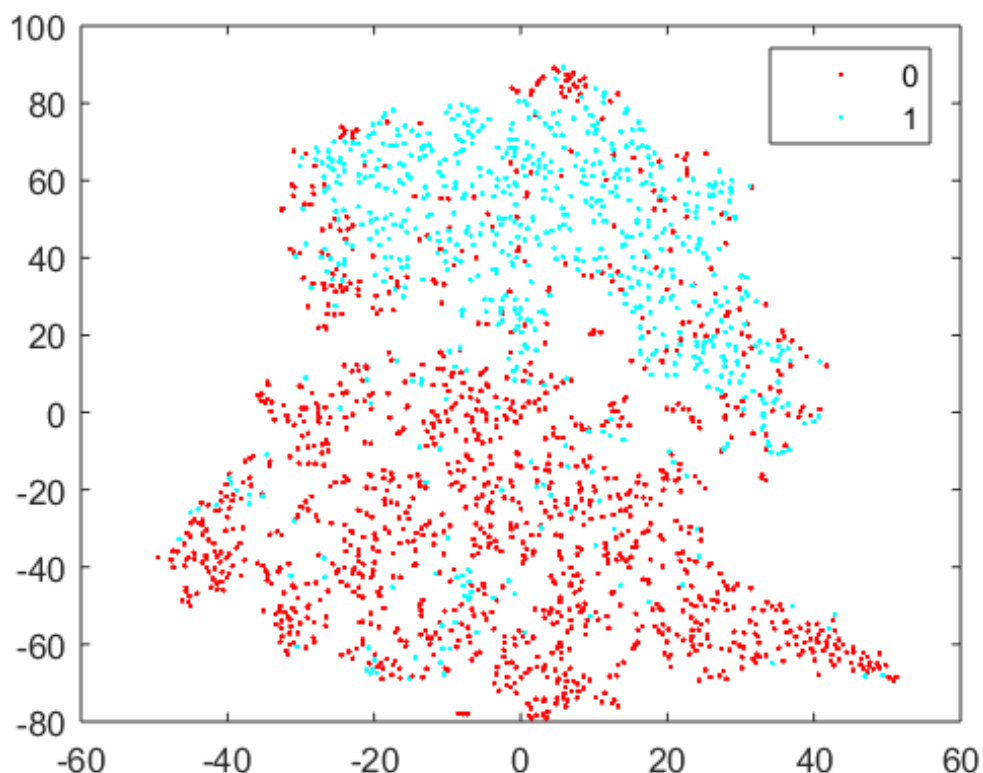


Рис. 5: обучающее множество в признаковом пространстве пониженной размерности

1.4. Постобработка. Шаг 2

Следующая проблема это окружности большого радиуса, которые содержат в себе несколько торцов бревен, которые также уже были найдены. Эта проблема характерна для многих фотографий, чтобы ее решить будем действовать следующим образом.

- Разобьем массив окружностей S_1 на два кластера с помощью алгоритма k-средних [6]. В качестве признакового описания окружностей будем использовать их радиусы.
- Будем считать, что подозреваемые на данную ошибку окружности выделились в отдельный кластер, если один из кластеров в два раза меньше другого и средний радиус по этому кластеру в два раза больше среднего радиуса по другому кластеру. Отметим этот кластер меткой 0, другой кластер - меткой 1.

- Если условия из предыдущего пункта выполнены, удалим из рассмотрения те элементы кластера 0, которые содержат в себе элементы кластера 1. Полученный массив обозначим C_2 .

Результат данного шага постобработки можно увидеть на рисунке 6.



Рис. 6: исходное изображение с найденными на нем окружностями, выполнены шаги постобработки 1, 2

1.5. Постобработка. Шаг 3

На последнем шаге постобработки удалим из массива C_2 те окружности, которые были классифицированы как торцы бревен, но не относятся к исследуемой пачке. Такие объекты могут появляться как в случае неправильной классификации на шаге 1 постобработки, так и в случае наличия на фотографии торцов бревен, не подлежащих анализу. Будем исходить из предположения, что анализируемых объектов на фотографии подавляющее большинство, и они сгруппированы с одинако-

вой плотностью в некоторой части фотографии. Для того чтобы определить эти объекты, воспользуемся алгоритмом плоскостной кластеризации DBCSCAN (density-based spatial clustering of applications with noise) [11]. Этот метод подходит для данной задачи, так как с помощью него можно разбить выборку на заранее неизвестное число кластеров произвольной формы.

Классический алгоритм DBCSCAN предназначен для кластеризации множества точек и использует евклидову метрику для нахождения расстояния между объектами.

$$d(p, q) = \sqrt{\sum_{k=1}^n (p_k - q_k)^2}$$

Так как по условию поставленной нами задачи необходимо разбить на кластеры множество окружностей в пространстве, необходимо выбрать подходящую метрику. В качестве такой метрики между двумя окружностями R_1 и R_2 выберем кратчайшее расстояние между множествами в евклидовом пространстве.

$$d(R_1, R_2) = \inf\{d(x_1, x_2), x_1 \in R_1, x_2 \in R_2\}$$

Очевидно, что рассчитать эту метрику легко по следующей формуле.

$$d(R_1, R_2) = d(c_1, c_2) - (r_1 + r_2)$$

Где c_1, c_2 - центры окружностей, а r_1, r_2 - радиусы этих окружностей. Геометрическое обоснование данной формулы приведено на рисунке 7.

Для поиска кластеров алгоритм DBSCAN считает количество точек в ϵ -окрестности каждого объекта p , и если их больше чем $MinPts$, создаётся новый кластер с корневым объектом p . При этом ϵ и $MinPts$ подаются на вход алгоритму. В представляемом алгоритме значение ϵ выбирается как среднее арифметическое радиусов окружностей из массива \mathbf{C}_2 , а $MinPts$ устанавливается равным 2.

Окружности принадлежащие наибольшему по численности кластеру выделяются в массив \mathbf{C}_3 , из которого далее формируется описанный

ранее выходной массив **A**. Результат данного шага постобработки можно увидеть на рисунке 8.

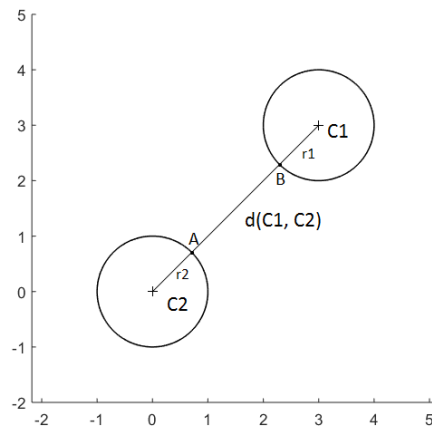


Рис. 7: геометрическое обоснование формулы расчета метрики между окружностями.



Рис. 8: исходное изображение с найденными на нем окружностями, выполнены шаги постобработки 1, 2, 3.

Глава 2. Используемые алгоритмы

2.1. Приведение цветного изображения к полутоновому

Полутоновое изображение — это изображение, пикселы которого представлены значениями яркости и соответствуют оттенкам серого. Полутоновое изображение кодируется в цифровом виде с помощью битовой карты. Каждый пиксел полутонового изображения может кодироваться с помощью 8 бит, что определяет количество возможных полутонов равным 256.

RGB — аддитивная цветовая модель, описывающая способ синтеза цвета. Изображение в данной цветовой модели состоит из трёх каналов: R - красный, G - зеленый, B - синий. Аддитивной она называется потому, что цвета получаются путём добавления к чёрному цвету основных цветов с различными коэффициентами интенсивности. Выбор этих цветов обусловлен особенностями физиологии восприятия цвета сетчаткой глаза человека.

Для перехода от изображения в цветовом пространстве RGB к полутоновому необходимо рассчитать интенсивность каждого пиксела $G_{i,j}$ следующей по формуле.

$$G_{i,j} = 0.299R_{i,j} + 0.587G_{i,j} + 0.114B_{i,j}$$

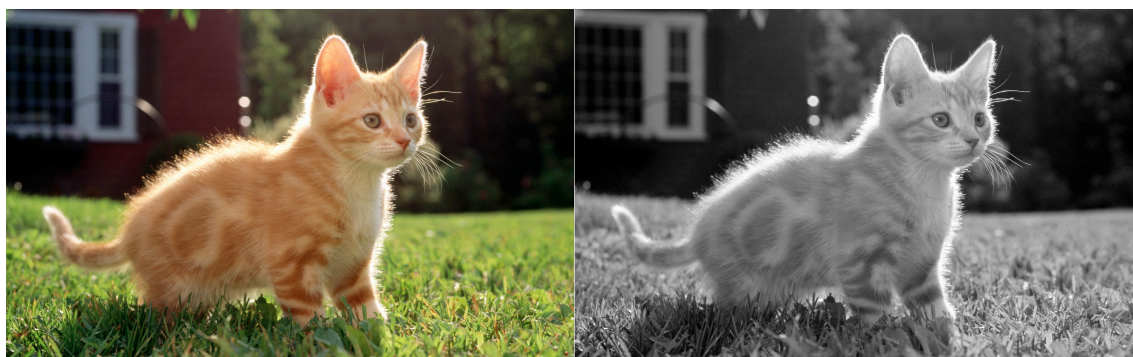


Рис. 9: цветное и полутоновое изображения.

2.2. Медианный фильтр

Медианный фильтр — нелинейный КИХ-фильтр, широко используемый в цифровой обработке изображений для подавления шумовых помех. Медианная фильтрация — эффективный способ удаления импульсных помех, таких как "соль перец". Окно фильтра перемещается по обрабатываемому изображению, и в каждой его точке производятся следующие вычисления. Значения отсчетов внутри окна фильтра сортируются в порядке возрастания. Значение, находящееся в середине упорядоченного списка, поступает на выход фильтра. Это значение присваивается соответствующей точке выходного изображения. Если количество отсчетов четное, то выходное значение фильтра в окне равно среднему значению двух отсчетов в середине упорядоченного списка. На рисунке 10 изображен пример работы медианного фильтра.

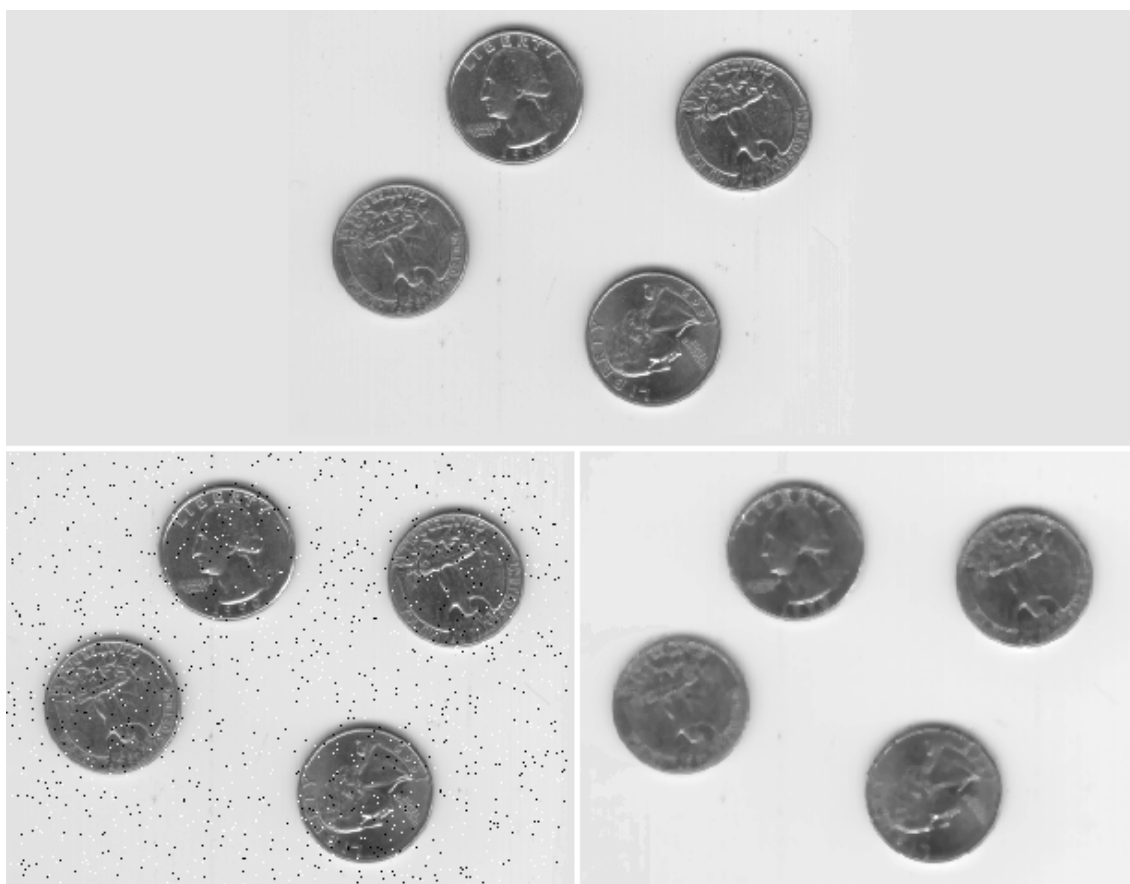


Рис. 10: оригинальное изображение (сверху), зашумленное изображение (слева), зашумленное изображение обработанное медианным фильтром (справа).

2.3. Свертка изображения с ядром

Пусть \mathbf{I} - изображение, представленное матрицей размерности $m \times n$, а \mathbf{G} - матрица размерности $k \times l$, называемая ядром. Сверткой изображения \mathbf{I} с ядром \mathbf{G} будем называть операцию получения матрицы $\mathbf{C} = \mathbf{I} * \mathbf{G}$, размерностью $m \times n$, элементы которой вычисляются следующим образом.

$$(\mathbf{I} * \mathbf{G})[m, n] = \sum_{k, l} I[m-k, n-l]g[k, l]$$

Визуализация свертки изображения с ядром приведена на рисунке 11.

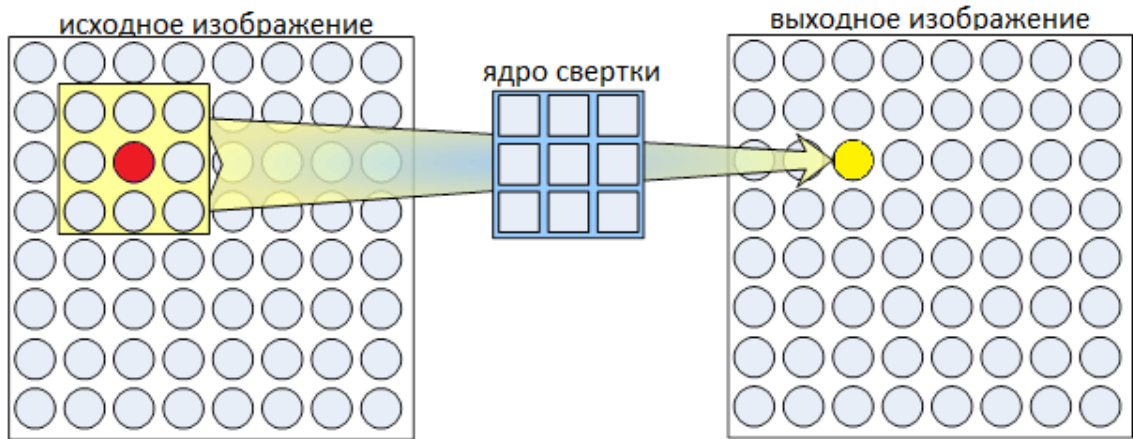


Рис. 11: свертка изображения с ядром.

Заметим, что невозможно рассчитать такие значения для элементов на границах изображения. Ширина этих границ будет равна $\lfloor \frac{k}{2} \rfloor$ сверху и снизу и $\lfloor \frac{l}{2} \rfloor$ слева и справа. Существует несколько методик заполнения граничных пикселей изображения \mathbf{C} . При использовании самой простой из них, значения элементов на границах матрицы \mathbf{C} берутся равными соответствующим элементам матрицы \mathbf{I} .

2.4. Фильтр Гаусса

Фильтр Гаусса является одним из фильтров пространственного сглаживания, то есть может быть рассмотрен как фильтр низких частот.

Применяется для очистки изображения от шума или достижения эффекта размытия изображения. Гауссовская фильтрация может быть реализована путем свертки изображения с ядром \mathbf{G} с размерностью $m \times n$. Центральный элемент ядра этого фильтра имеет максимальное значение, соответствующее пику распределения Гаусса (нормального распределения). Импульсная характеристика определена на \mathbf{G} , а элементы этой характеристики описываются следующим выражением.

$$h(i, j) = \frac{h_g(i, j)}{\sum_{i=1}^m \sum_{j=1}^n h_g(i, j)}, \quad h_g(i, j) = e^{\frac{-(i^2+j^2)}{2\sigma^2}}$$

Выражение $h_g(i, j)$ определяет двумерное нормальное распределение со среднеквадратическим отклонением σ . Пример работы фильтра гаусса приведен на рисунке 12.



Рис. 12: оригинальное изображение (слева), зашумленное изображение (справа), зашумленное изображение обработанное фильтром Гаусса (снизу).

2.5. Метод гамма-коррекции

Пусть исходное изображение I имеет один из следующих дефектов: узкий или смещенный диапазон яркостей пикселей, концентрация яркостей вокруг определенных значений, неравномерное заполнение диапазона яркостей. Тогда к изображению необходимо применить преобразование яркостей, компенсирующее нежелательный эффект.

$$f^{-1}(y) = x$$

В данной формуле y - яркость пиксела на исходном изображении, x - яркость пиксела на выходном изображении.

Одной из часто применяемых функций является функция гамма-коррекции. Параметры s и γ являются настраиваемыми.

$$y = cx^\gamma$$

Для иллюстрации работы данного метода введем понятие гистограммы изображения. Гистограмма — это график, в котором по горизонтальной оси представлена яркость, а по вертикали — относительное число пикселей с конкретным значением яркости. Пример работы алгоритма гамма-коррекции приведен на рисунках 13,14.

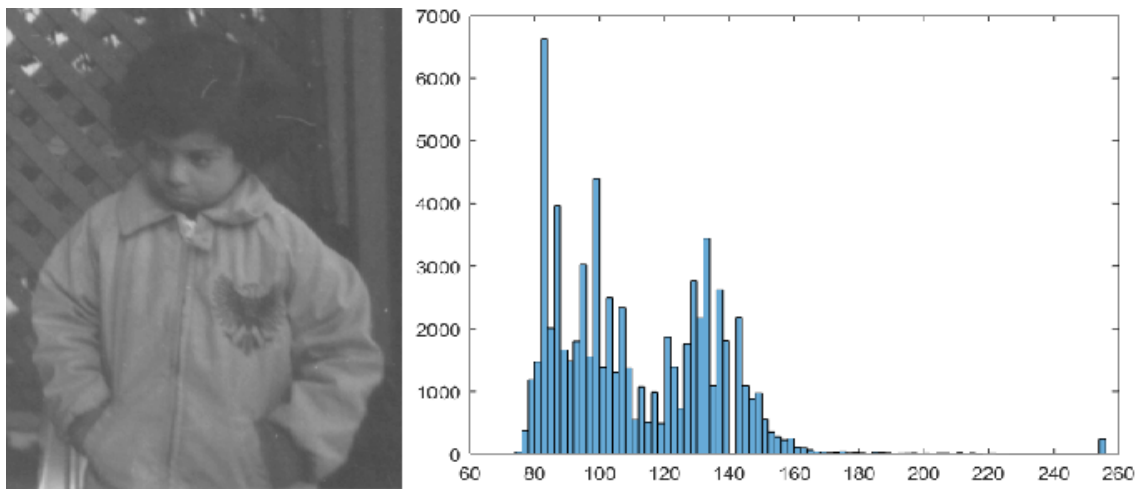


Рис. 13: оригинальное изображение и его гистограмма.

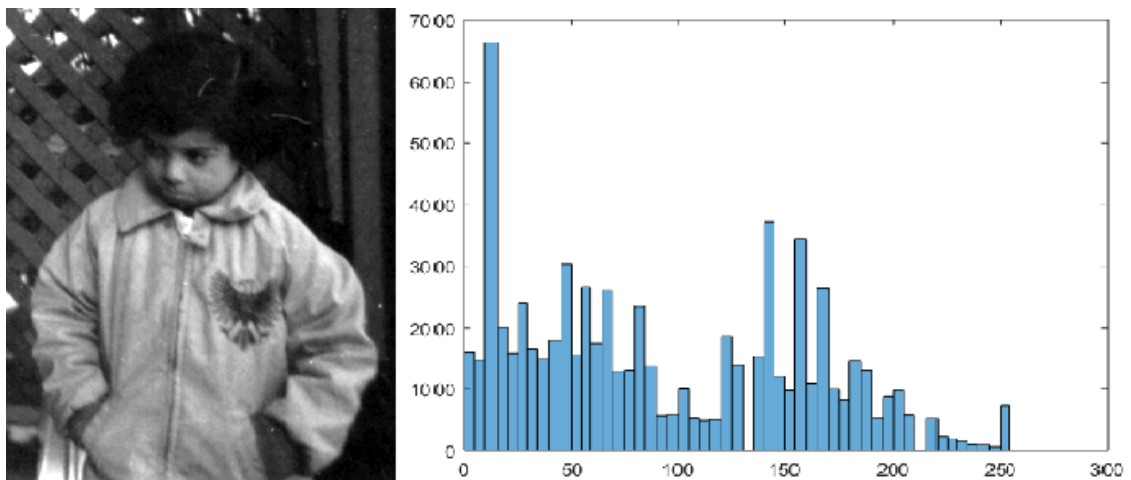


Рис. 14: контрастированное изображение и его гистограмма.

2.6. Детектор границ Канни

Оператор Канни [3] — оператор используемый для обнаружения границ на изображении. Был разработан в 1986 году Джоном Канни и является многоступенчатым алгоритмом нахождения контуров. Этот алгоритм в большинстве случаев наилучшим образом справляется с данной задачей и наиболее часто применяется по сравнению со своими аналогами. Альтернативой оператору Канни могут быть такие алгоритмы, как оператор Собеля, оператор Прюитт, оператор Лапласа.

Оператор Канни учитывает следующие три критерия оптимальности.

- Хорошее обнаружение. Это свойство трактовалось Канни как повышение отношения сигнал/шум.
- Хорошая локализация, то есть правильное определение положения границы.
- Единственный отклик на одну границу.

Алгоритм состоит из следующих пяти шагов. Пусть на вход подается полутоновое изображение \mathbf{I} размерности $n \times m$.

- Размытие изображения для удаления шума. Производится с помощью фильтра Гаусса.
- Вычисляется матрица \mathbf{G} размерности $n \times m$ приближенных значений градиентов яркости изображения в каждой его точке. Эта операция производится с помощью оператора Собеля, который можно записать следующим образом. Пример работы оператора Канни приведен на рисунке 15.

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{I} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{I}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

- Только локальные максимумы G рассматриваются как потенциальные границы.
- Производится двойная пороговая фильтрация. Потенциальные границы разделяются двумя порогами на три группы: не границы, слабые границы, сильные границы.
- Производится трассировка области неоднозначности. Итоговые границы определяются следующим путем. Берутся все сильные границы, и к ним добавляются слабые границы, связанные с сильными.

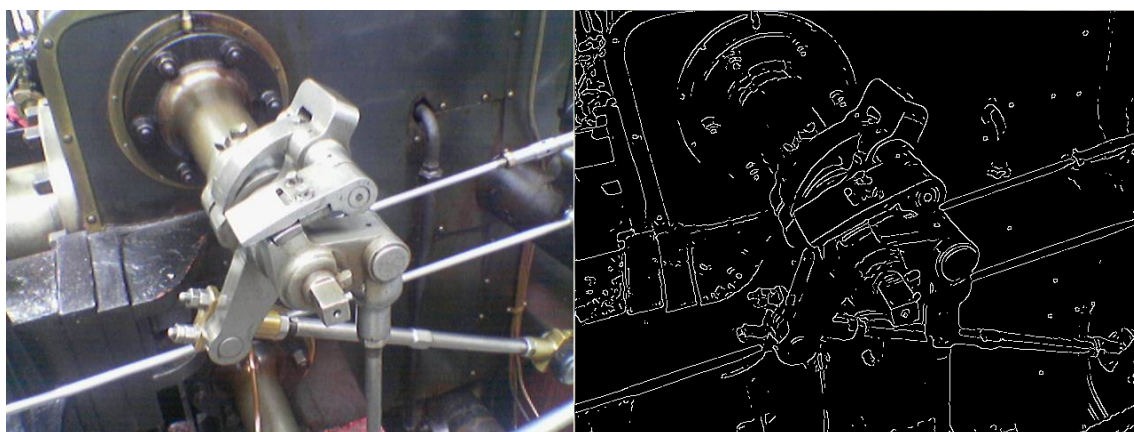


Рис. 15: оригинальное изображение (слева), изображение обработанное оператором Канни(справа).

2.7. Алгоритм Хафа для поиска окружностей

Алгоритм Хафа - метод для нахождения на изображении аналитических кривых. В данной работе будут описаны модификации данного алгоритма предназначенные для обнаружения окружностей заданного или переменного радиуса. Будем предполагать, что на вход поступает полутоновое изображение. Основной идеей алгоритма является использование оценки ориентации нормали в голосующих контурных точках.

Будем считать, что имеем двухпараметрическое семейство кривых,

$$(x-x_0)^2 + (y-y_0)^2 = R^2$$

и производить поиск максимума аккумуляторной функции (массива) $\mathbf{A}(x, y)$.

Первым шагом алгоритма будет обнаружение на изображении точек, относящихся к присутствующим на нём контурам. Для этого будем использовать рассмотренный ранее оператор Канни. Голосующими контурными точками считаются те точки, которые были определены оператором Канни как границы объектов.

Далее для всех найденных краевых пикселей находится оценка положения и ориентации контура, то есть строится касательная и нормаль к нему. Затем производится движение на расстояние R от краевого пикселя в направлении нормали к контуру. Это делается для нахождения центра кругового объекта радиуса R . Если такую операцию проводить для каждого пикселя принадлежащего границе, будет найдено множество положений предполагаемых точек центра. Эти значения необходимо занести в аккумуляторный массив, то есть для всех предполагаемых точек центра инкрементировать значение $\mathbf{A}(x_0, y_0)$, где (x_0, y_0) предполагаемая точка центра. Для определения местонахождения центра, необходимо усреднить найденные предполагаемые значения точек центра. Если окружностей на изображении несколько, то предполагаемые значения будут локализованы вокруг точек их центров. Для того, чтобы их найти, необходимо нанести на аккумуляторный массив $\mathbf{A}(x, y)$ сетку выбранного размера и усреднить значения внутри её ячеек, получив тем самым массив меньшей размерности. Находя локальные максимумы этого массива мы будем находить координаты центров искомых окружностей.

Для случаев когда радиус окружности является переменным, необходимо включить R в качестве дополнительной неизвестной величины в параметрическое пространство-аккумулятор. Тогда процедура поиска локального экстремума должна определить радиус, так же как и место нахождения центра путем рассмотрения изменений вдоль третьего

измерения параметрического пространства. Пример работы алгоритма Хафа изображен на рисунке 16.

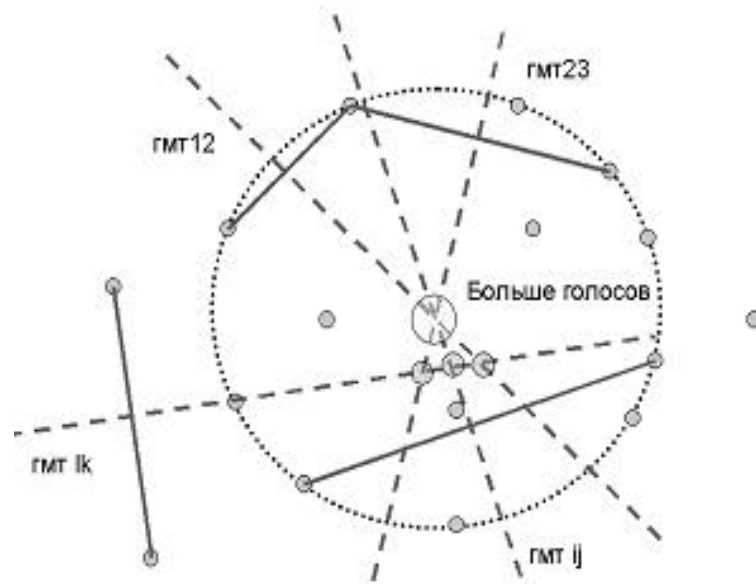


Рис. 16: обнаружение окружности неизвестного радиуса в бинарном точечном множестве.

2.8. Цветовое пространство LAB

LAB - цветовое пространство, широко применяемое при анализе изображений. При разработке LAB ставилась задача создания такого цветового пространства, изменение цвета в котором будет более линейным для человеческого восприятия. Это значит, что одинаковое изменение значений координат цвета в разных областях цветового пространства должно производить одинаковое ощущение изменения цвета у человека.

В цветовом пространстве LAB значение яркости отделено от значения хроматической составляющей цвета, то есть тона и насыщенности. Яркость задана координатой L и изменяется от 0 до 100. Хроматическая составляющая представлена двумя декартовыми координатами A и B, где первая обозначает положение цвета в диапазоне от зеленого до красного, вторая — от синего до желтого.

Благодаря выделению яркостной составляющей в отдельную координату цветное пространство LAB широко применяется при анализе изображений. Существует возможность при помощи данного цветового пространства распознавать объекты по цвету вне зависимости от того, при каком освещении велась съемка. Для этого обычно строят гistogramмы изображения по координатам А и В и включают их в признаковое представление изображения. Структура цветового пространства LAB представлена на рисунке 17.

Переход от пространства RGB к LAB осуществляется следующим образом.

$$r = \frac{R}{255}, \quad g = \frac{G}{255}, \quad b = \frac{B}{255}$$

$$\begin{vmatrix} l \\ a \\ b \end{vmatrix} = \frac{\ln \left(\begin{vmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.8444 & 0.1288 \end{vmatrix} \times \begin{vmatrix} r \\ g \\ b \end{vmatrix} \right)}{\ln(10)}$$

$$MP = \begin{vmatrix} 0.574 & 0 & 0. \\ 0 & 0.4082 & 0 \\ 0 & 0 & 0.7071 \end{vmatrix} \times \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{vmatrix}$$

$$\begin{vmatrix} L \\ A \\ B \end{vmatrix} = MP \times \begin{vmatrix} l \\ a \\ b \end{vmatrix}$$

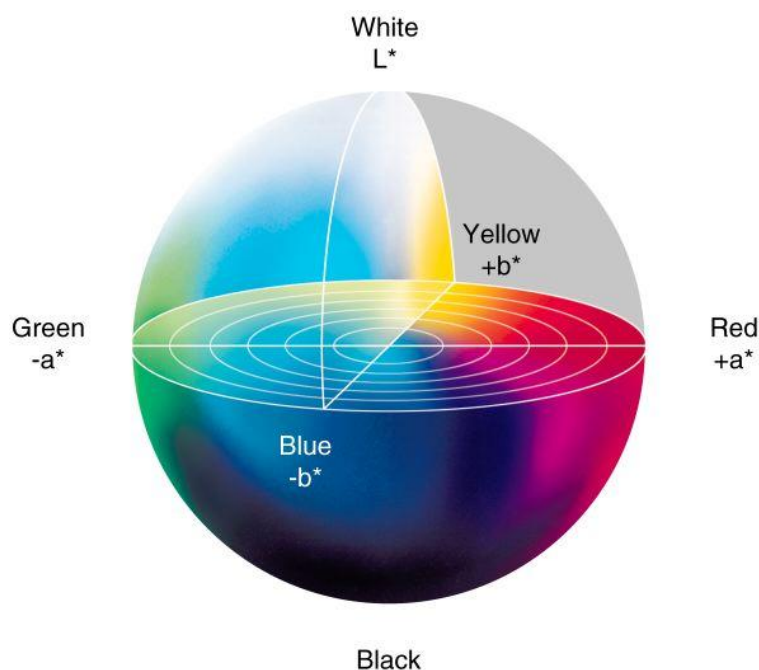


Рис. 17: структура цветового пространства LAB.

2.9. Сегментационный фрактальный анализ текстур

Алгоритм сегментационного фрактального анализа предназначен для построения признакового представления текстуры изображения **I**. На вход алгоритму подается полутоновое изображение и некоторый параметр n_t , значение которого будет раскрыто далее.

На первом шаге необходимо сегментировать изображение по уровням яркости. Для поиска порогов сегментации воспользуемся многоуровневым алгоритмом Отцу [5]. Идея алгоритма Отцу заключается в том, чтобы разбить изображение на две области так, чтобы минимизировать внутриобластную дисперсию. Применяя этот алгоритм рекурсивно, получим множество пороговых значений. Количество искомых порогов обозначается n_t и является входным параметром алгоритма. Множество порогов T дополним значением самого яркого пиксела на изображении $T = \{t_i\} \cup n_l$. Далее с помощью найденных пороговых значений получим множество бинарных изображений $\{\mathbf{B}_i\}$.

$$\mathbf{B}_i(x, y) = \begin{cases} 1, & \text{if } \mathbf{I}(x, y) \geq t_i \\ 0, & \text{else} \end{cases}$$

Для всех изображений из $\{\mathbf{B}_i\}$ построим контурные изображение $\{\mathbf{G}_i\}$. В данной формуле $N_8[(x, y)]$ - множество пикселей, находящихся в отношении 8-связности с (x, y) . По полученным изображениям найдем следующие величины.

$$\mathbf{G}_i(x, y) = \begin{cases} 1, & \text{if } \exists (\bar{x}, \bar{y}) \in N_8[(x, y)] : B_i(\bar{x}, \bar{y}) = 0 \wedge B_i(x, y) = 1 \\ 0, & \text{else} \end{cases}$$

- $A_j = \frac{N_l}{N}$, где N_l – количество пикселей со значением 1 на изображении \mathbf{B}_j , N – общее количество пикселей этом изображении.
- v_j - средний уровень серого пикселей изображения \mathbf{I}_i , соответствующих пикселям со значением 1 изображения \mathbf{B}_i .
- N_{8j} - количество 8-связных областей на изображении \mathbf{B}_i . Эту величину можно рассчитать с помощью известного алгоритма, позволяющего осуществлять маркировку изображений [10]. Следует отметить, что в авторском описании сегментационного фрактального анализа данный пункт отсутствует. Он был введен в алгоритм данной работе, так как это дало заметные улучшения результатов классификации.
- D_j – фрактальная размерность [8], вычисленная по изображению \mathbf{B}_i с помощью метода box-counting [8]. Это наиболее важная характеристика. Она описывает сложность линий контуров сегментированных областей.

Структура алгоритма сегментационного фрактального анализа представлена на рисунке 18.

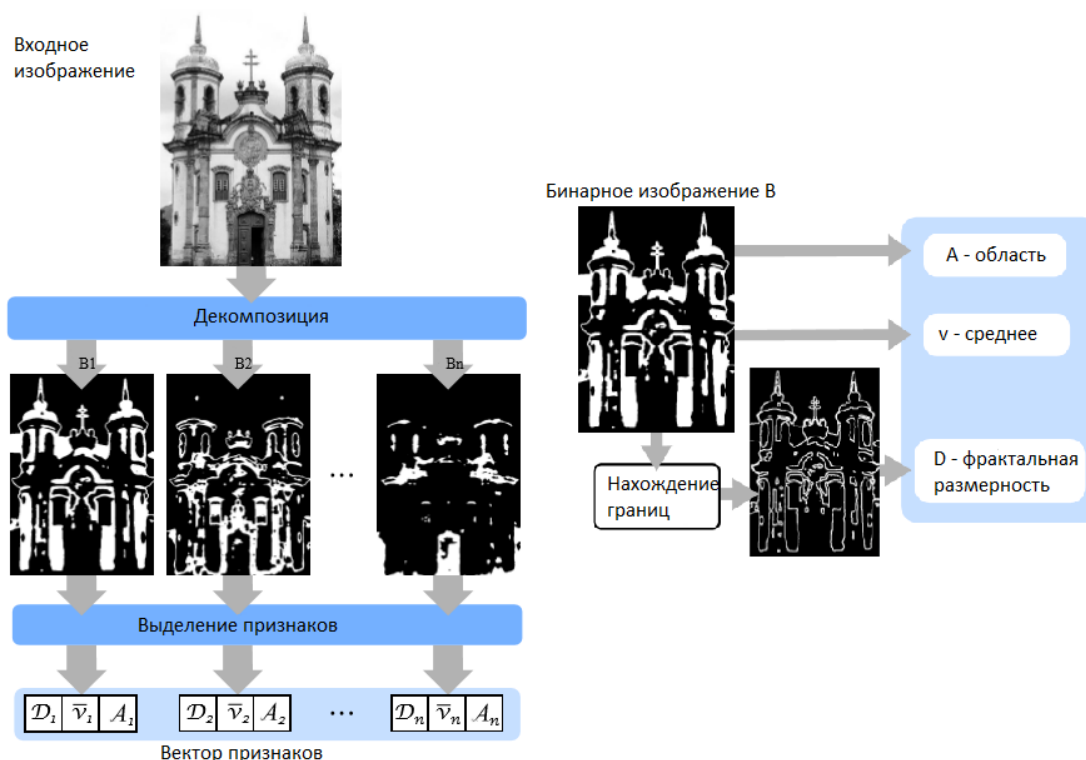


Рис. 18: структура алгоритма сегментационного фрактального анализа.

2.10. Метод опорных векторов

Метод опорных векторов является одним из наиболее популярных методов решения задачи машинного обучения с учителем. Был предложен В. Н. Вапником и известен в англоязычной литературе под названием SVM (Support Vector Machine). Относится к семейству линейных классификаторов.

Приведем постановку задачи машинного обучения с учителем. Пусть имеется множество признаковых представлений объектов \mathbf{X} и множество возможных ответов \mathbf{Y} . Под признаковыми представлениями объектов подразумеваются вектора некоторой длины, определяемой по своему для каждой конкретной задачи. Существует некоторая неизвестная зависимость между ответами и объектами. Однако известно конечное множество прецедентов — пар «объект, ответ», называемых обучающей выборкой $X^m = \{(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)\}$. На основе этих данных требуется построить алгоритм, который для любого произвольного объекта может выдать достаточно точный ответ.

Идея алгоритма SVM состоит в построении в некотором смысле оп-

тимальной разделяющей гиперплоскости, то есть обученный классификатор будет представлять собой вектор коэффициентов $\lambda = \lambda_1, \dots, \lambda_n$, где n длина вектора признаков объекта. Оптимальная разделяющая гиперплоскость представлена на рисунке 19. Знак скалярного произведения вектора признаков x на вектор коэффициентов λ и будет ответом классификатора на объекте x . Под оптимальностью гиперплоскости в данном случае понимается наибольшая её удаленность от объектов обучающего множества. Задача обучения решается путем оптимизации функционала метода опорных векторов. Этот функционал представляет собой сумму функционала, аппроксимирующего функционал эмпирического риска кусочно линейной функцией, и функционала нормы вектора коэффициентов λ .

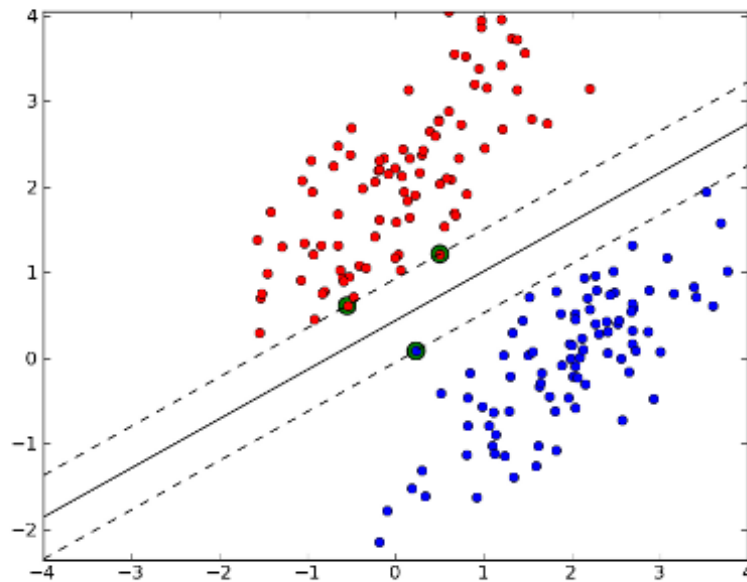


Рис. 19: оптимальная разделяющая гиперплоскость в двумерном пространстве.

2.11. Кластеризация методом k-средних

Метод k-средних — один из наиболее популярных методов кластеризации. На вход алгоритму подается множество точек \mathbf{X} и количество кластеров k , на которые их надо разбить. Алгоритм стремится мини-

минимизировать суммарное квадратичное отклонение точек кластеров от центров их кластеров.

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

В данной формуле k — число кластеров, S_i — полученные кластеры, $i = 1, 2, \dots, k$ и μ_i — центры масс векторов $x_j \in S_i$.

Алгоритм завершается, когда на следующей итерации не изменяется центр масс кластеров. Это происходит за конечное число итераций, потому что число разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение V не увеличивается, так что заикливание невозможно. На рисунке 20 представлены результаты работы данного алгоритма.

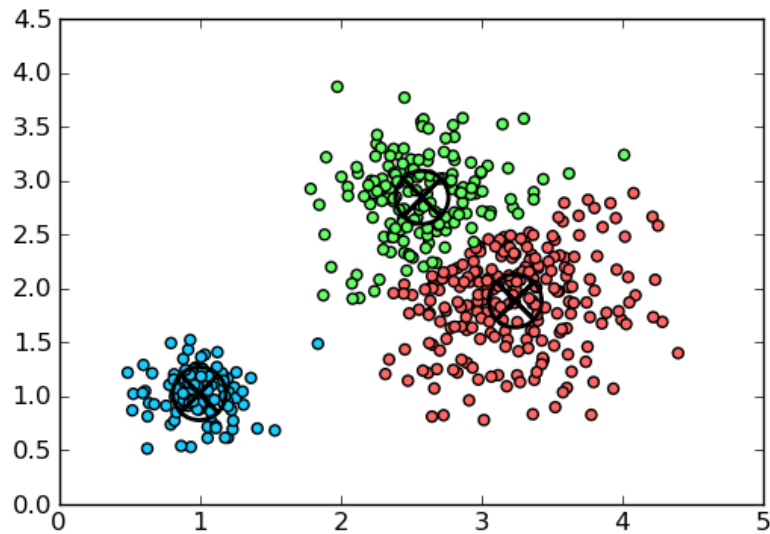


Рис. 20: разделение множества точек на плоскости на три кластера.

2.12. Алгоритм DBSCAN

Алгоритм DBSCAN был разработан Мартином Эстер, Гансом-Питером Кригель и коллегами в 1996 году как решение проблемы разбиения данных на кластеры произвольной формы. Является алгоритмом плотностной кластеризации. Подавляющее количество алгоритмов, произ-

водящих кластеризацию, выделяют кластеры по форме близкие к сферическим, так как минимизируют расстояние от объектов до центра кластера. Было экспериментально установлено, что алгоритм DBSCAN способен распознать кластеры любой формы.

В основу алгоритма положена идея, заключающаяся в том, что объекты каждого кластера распределены с типичной плотностью, которая заметно выше, чем плотность вне кластера. Кроме того, плотность в областях с шумом существенно ниже плотности любого из кластеров. Это означает, что для каждой точки кластера её окрестность заданного радиуса должна содержать не менее заданного числа точек. Это число задаётся пороговым значением *MinPts*, которое подается на вход. Радиус окрестности также подается на вход, обозначим его ϵ .

Для поиска кластеров алгоритм DBSCAN проверяет ϵ -окрестность всех объектов. Расстояние между точками рассчитывается с помощью Евклидовой метрики. Если ϵ -окрестность объекта p содержит больше точек чем *MinPts*, то создаётся новый кластер с корневым объектом p . Затем DBSCAN итеративно ищет точки непосредственно плотно-достижимые из корневых объектов. Эти точки могут добавляться в данный кластер. Процесс завершается, когда ни к одному кластеру не может быть добавлено ни одного нового объекта. Объекты не отнесенные ни к одному кластеру считаются шумовыми. На рисунке 21 представлены результаты работы данного алгоритма для некоторого распределения точек на плоскости.

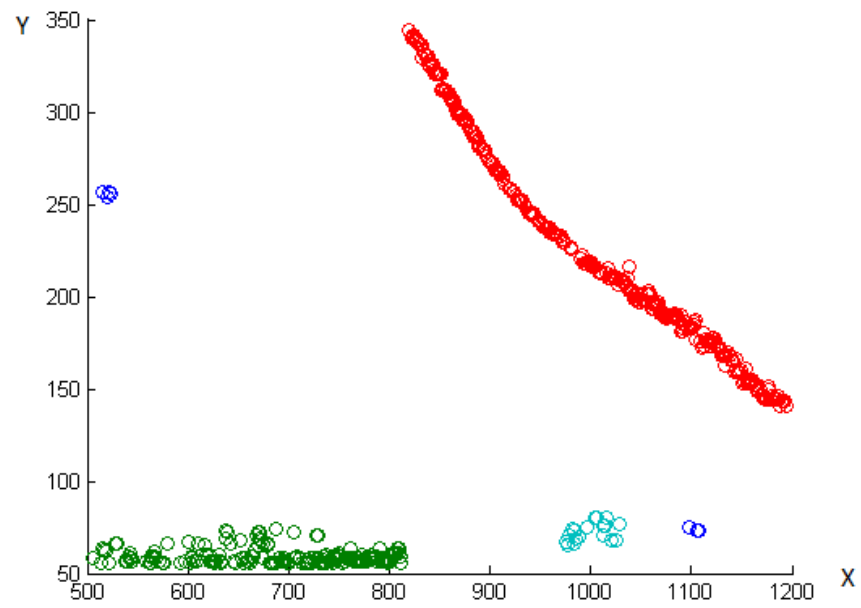


Рис. 21: разделение множества точек на плоскости с помощью алгоритма DBSCAN.

Глава 3. Программная реализация и результаты тестирования

Для проведения исследования была написана программа-прототип в среде разработки MATLAB 2015a. Были применены встроенные пакеты Image Processing Toolbox и Machine Learning Toolbox. Тестирование работы программы проводилось на ноутбуке под ОС Windows 10 с процессором Intel Core i7-3517U 1,9 ГГц.

В ходе тестирования с помощью программы было исследовано 5 фотографий лесовозов, на которых в общей сложности было 546 бревен. Из них 468 было успешно обнаружено, а 78 было пропущено программой, то есть количество правильно найденных бревен составило 85.7% от их общего числа. Кроме того, было допущено 16 ложных обнаружений, таким образом количество ложных срабатываний составляет 3.3% от всех обнаружений. Итак, на тестовых данных была показана работоспособность разработанного алгоритма.

На рисунке 22 будет представлена визуализация структуры разработанного алгоритма.

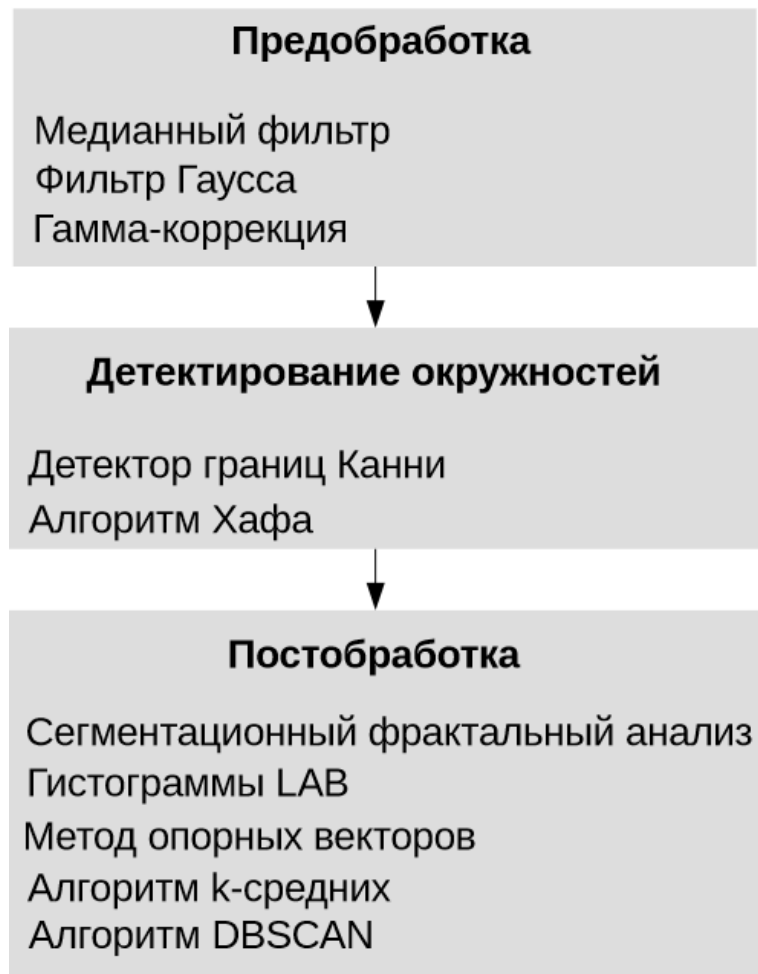


Рис. 22: визуализация структуры разработанного алгоритма.

Выводы

Исходя из результатов тестирования можно заключить, что предложенный алгоритм успешно справляется с поставленной задачей. Несмотря на это, стоит заметить, что при его применении часть искомых объектов все таки остается необнаруженной. Дополнительные исследования показали, то это связано с ошибками на этапе поиска окружностей на изображении, а именно с недостаточной точностью обнаружения контуров на изображениях. Необходимо отметить, что используемые в исследовании примеры изображений изначально не предназначались для извлечения из них той информации, получение которой было целью данной работы. Они были сделаны для учета номерных знаков лесовозов. Поэтому есть причины предполагать, что при использовании данного алгоритма на практике возможно существенное улучшение качества фотографирования. Это в свою очередь приведет к улучшению качества распознавания границ на изображениях и улучшению результатов работы программы.

Кроме того, обучающая выборка, представленная для решения задачи машинного обучения с учителем на первом шаге постобработки, недостаточно велика по сравнению с обучающими выборками, используемыми в реальных задачах. Велик шанс того, что при использовании более объемной обучающей выборки качество работы алгоритма улучшится.

Заключение

В ходе данного исследования были достигнуты следующие результаты.

- Разработан алгоритм предобработки изображения, положительным образом влияющий на качество работы алгоритма распознавания образов.
- Предложено возможное решение для задачи машинного обучения с учителем, где объектами являлись фрагменты изображения. Был разработан алгоритм извлечения признаков из изображения. В ходе этой работы было предложено усовершенствование алгоритма сегментационного фрактального анализа.
- Предложено решение для задачи пространственной кластеризации с неизвестным количеством кластеров. Кроме того, объектами, которые подлежали кластеризации, были окружности. Для решения этой задачи был применен алгоритм DBSCAN, а так же предложена его модификация для работы с окружностями.

Список литературы

- [1] Atherton T.J., Kerbyson D.J. Size invariant circle detection // Image and Vision Computing. — 1999. — 9. — P. 795–803.
- [2] Borovickr J. Circle Detection Using Hough Transforms Documentation // COMS30121 - Image Processing and Computer Vision. — 2003. — 4.
- [3] Canny J. A Computational Approach To Edge Detection // IEEE Trans. Pattern Analysis and Machine Intelligence. — 1986. — P. 679–698.
- [4] Geusebroek J., Smeulders Arnold, van de Weijer Joost. Fast Anisotropic Gauss Filtering // IEEE Transactions on Image Processing. — 2003. — 8. — P. 938 – 943.
- [5] Geusebroek J., Smeulders Arnold, van de Weijer Joost. An Efficient Algorithm for Fractal Analysis of Textures // 25th SIBGRAPI Conference on Graphics, Patterns and Images. — 2012. — 8. — P. 39 – 46.
- [6] Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. — 2 edition. — Springer, 2009. — Vol. 699.
- [7] Lee Po-Ming, Chen Hung-Yi. Adjustable gamma correction circuit for TFT LCD // IEEE International Symposium on Circuits and Systems. — 2005. — 5. — P. 780 – 783.
- [8] Long M., Peng F. A Box-Counting Method with Adaptable Box Height for Measuring the Fractal Feature of Images // Radioengineering. — 2013. — 4. — P. 208–213.
- [9] Perreault S., Hebert P. Median Filtering in Constant Time // IEEE Transactions on Image Processing. — 2007. — 9. — P. 2389 – 2394.

- [10] Suzuki S., Abe K. Topological Structural Analysis of Digitized Binary Images by Border Following // Computer vision, graphics, and image processing. — 1985. — P. 32–46.
- [11] A density-based algorithm for discovering clusters in large spatial databases with noise / M. Ester, H.-P. Kriegel, J. Sander, X. Xu // Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining. — 1996. — P. 226–231.
- [12] van der Maaten L. Visualizing Data using t-SNE // Journal of Machine Learning Research. — 2008. — 9. — P. 2579–2605.

Приложение

На рисунках 23-27 представлена визуализация результатов работы описанного алгоритма.



Рис. 23: результат работы алгоритма.



Рис. 24: результат работы алгоритма.



Рис. 25: результат работы алгоритма.



Рис. 26: результат работы алгоритма.



Рис. 27: результат работы алгоритма.